Community Detection on Ego Networks via Mutual Friend Counts

Nicolas Kim
Department of Statistics
Carnegie Mellon University
Pittsburgh, PA 15213
nicolask@stat.cmu.edu

Alessandro Rinaldo
Department of Statistics
Carnegie Mellon University
Pittsburgh, PA 15213
arinaldo@cmu.edu

Abstract

Community detection is an important tool for understanding the structure of social networks, and spectral clustering is the standard algorithm for performing community detection within the statistics literature. However, its polynomial-order runtime means it cannot be applied to contemporary social networks with billions of nodes. To address this computational challenge we provide a statistical analysis of a simple algorithm that solves a more tractable problem: community detection for the ego network. Under a broad class of parametrizations for the stochastic block model on the data, this algorithm recovers the correct community labels for the given ego network. Simulations show that this algorithm outperforms spectral clustering when the number of communities is large.

1 Introduction

Today's social networks are on track to encompass the world's population. Facebook claims to have 1.79 billion active monthly active users, a larger figure than the combined population of China and the United States of America. These massive networks present an opportunity to understand social connectivity on an unprecedented scale. Of particular interest to social networking companies is the potential to add compelling features for users and for advertisers. Clustering users is one way of extracting relevant information from the network; however, these networks evolve rapidly, which necessitates constant updating of these clusters in order to keep user and ad metrics as relevant as possible.

The computational challenge of clustering networks with several billions of nodes has been unmet among algorithms with statistical performance guarantees. Clustering for the stochastic block model (SBM) has been thoroughly studied. [1, 19]; the canonical approach is to perform spectral clustering on some form of the graph Laplacian. Theoretical results by [9, 14] establish statistical consistency for this algorithm, but it is $O(|V|^3)$ and thus not suited for clustering networks with billions of nodes. Approximations to the community detection task have been proposed, notably in [18] with statistical guarantees based on a pertubation analysis, but this approximation is still unable to cluster a network with billions of nodes. Spectral clustering has been shown to be parallelizable by [7], and very recent results can cluster a billion-node graph on the order of several hours [17] using a label-propagation method, but cluster quality has not been shown to be statistically guaranteed in either case.

Given the apparent difficulty of simultaneously satisfying all three criteria (clustering the whole network; with statistical quality guarantees; quickly), we can now introduce a class of methods which takes a different approach by forgoing the first condition of clustering the whole network. Local methods for network clustering are a recent breed of algorithms which are inspired by the idea that one rarely needs to cluster an entire billion-node network all at once. Rather, the small neighborhood of a single node of interest should contain a lot of the information relevant to that particular node. The development of algorithms that can reliably and instantaneously determine the community structure around specific individuals in massive networks has been driven by [16, 4, 3, 12, 10, 15, 8]. These results have generally come from the theoretical computer science community, and as such, cluster quality comes in the form of guarantees on the volume of the identified partition, or the conductance of the cut. These are traditionally graph-theoretic properties for which there are currently no translations to properties of common statistical graph models, such as the SBM.

Our main contribution is the `Mutual-Friends` algorithm which can determine the set of nodes in the ego network which are likely to belong to the same community as the ego node. Statistical consistency is shown for this algorithm. Some of the results are analogous to those in [6], where the friend counts (rather than mutual friend counts) are shown to separate the network communities. Certain properties of the ego network make clustering easier, although the proofs require more subtle manipulations in comparison.

2 The `Mutual-Friends` Algorithm

In this section, we introduce Algorithm 1, the `Mutual-Friends` algorithm, to specifically address the problem of partitioning any given node's list of connections. This task corresponds to detecting which nodes in a given ego network belong to the same community as the ego node.[1] The more general task of finding communities within ego networks is tackled by [11], but they utilize node covariates and do not provide statistical guarantees. Our method only requires the degrees of the nodes in the ego network, using an approach adapted from [6], which estimates the community memberships of a whole network drawn from a SBM using only the degrees of the nodes. Our algorithm exploits the fact that in an SBM, it is usually the case that $\pi_{ii} \gg \pi_{ij}\ \forall j \neq i$; this makes the partition boundary considerably more learnable.

Algorithm 1: Mutual-Friends

```
Data: Ego node $e$; mutual friend counts $(D_i^*)_i$ for all neighbors $i \in$ neighbors$(e)$; and a
      one-dimensional clustering algorithm cluster that outputs integer classes
Result: Ideally, the maximal set of nodes $V_e$ such that $Z_i = Z_e$ and $A_{ie} = 1$ for all $i \in V_e$
begin
    $\hat{Z} \longleftarrow$ cluster$((D_i^*)_i)$, num_clusters $= 2)$;
    $m_0 \longleftarrow$ mean$((D_i^*)_{i:\hat{Z}_i=0}))$;
    $m_1 \longleftarrow$ mean$((D_i^*)_{i:\hat{Z}_i=1}))$;
    if $m_0 > m_1$ then
        return $\{v \in neighbors(e) : \hat{Z}_v = 0\}$;
    end
    else
        return $\{v \in neighbors(e) : \hat{Z}_v = 1\}$;
    end
end
```

We consider `Mutual-Friends` to be an *algorithmic primitive*, by which we mean a kind of subroutine for a more complicated function that iterates `Mutual-Friends`, similarly to [5]. In practice, having an algorithmic primitive which depends solely on mutual friend counts is appealing because this information is likely to be either cached or optimally computable by the graph-processing architectures of companies which host these large social networks. The probabilistic properties of `Mutual-Friends` are also more easily studied compared to algorithms like `Nibble` and `PageRank-Nibble`. A better understanding of these properties will simplify the derivation of similar properties for the more complex algorithms that iterate this primitive.

Figure 1 illustrates how the observed mutual friend count distribution is obtained from an example whole network, which is generated according to the following procedure:

[1]The *ego network* is the subgraph induced by including only the nodes that are connected to a chosen, fixed, *ego node*. If the whole network is $G = (V, E)$, we denote the ego node by $e \in V$, and its induced ego network is G_e. The neighbors of e are also referred to as *alters*.

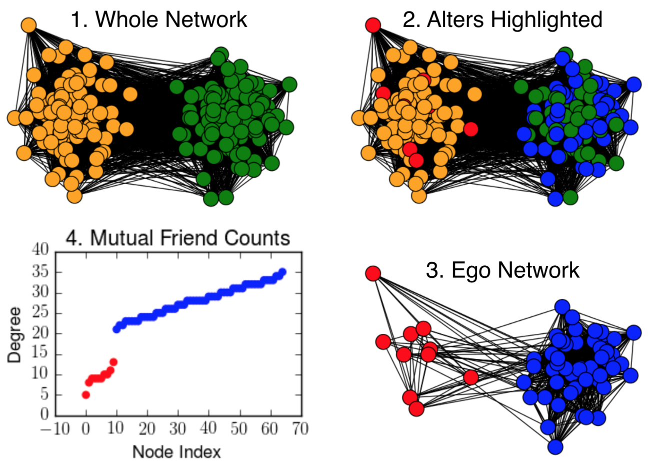

Figure 1: Even though the community structure in the whole network is totally symmetric, the ego network is unbalanced since nodes from the same community as the ego are more likely to be in the ego network.

1. The whole network is sampled from an SBM with parameters $\pi_{11} = \pi_{22} = 0.5$, $\pi_{12} = 0.1$, and $\alpha = (0.5, 0.5)$. This corresponds to two balanced and equivalent communities, denoted by orange and green.
2. An ego node is chosen, which happens to be green, and its alters are highlighted in red and blue, corresponding to the orange and green communities, respectively.
3. The ego network is the subgraph induced by the red and blue nodes.
4. The mutual friend counts correspond to the degrees of the nodes in the ego network.

2.1 Terminology

We consider the case of an undirected social network with n nodes, $G = (V, E)$. This network is drawn from an SBM with Q true communities, with the symmetric block tie probability matrix $[\pi_{ij}]_{1\le i,j\le Q}$. We let $(Z_1, \ldots, Z_n) \in [Q]^n$ denote the vector of community assignments, and $Z_i \overset{\text{iid}}{\sim} \text{Multinomial}(\vec{\alpha}; 1)$, where $\vec{\alpha} = (\alpha_1, \ldots, \alpha_Q) \in [0, 1]^Q$ such that $\sum_q \alpha_q = 1$ is the vector containing the probability any node belongs to each community. In other words, $\mathbb{P}(Z_i = q) = \alpha_q$ for all i and q. We assume that $\pi_{ii} > \pi_{ij}$ for all $j \neq i$; this condition is commonly referred to as assortativity.

We adopt a modification of the notation used by [6] for the ego network case. By way of comparison, recall that [6] denote the degree of a given node i by $D_i := \sum_{j\neq i} A_{ij}$. The normalized degree is $T_i := D_i/(n - 1)$. Then, the largest deviation of any node's normalized degree from its expected value is

$$d_n := \max_{q\in[Q]} \sup_{i\in[n]:Z_i=q} |T_i - \bar{\pi}_q|.$$

Note that these degrees are exchangable (but not independent). They can be thought of as being sampled from a mixture distribution, with the mean of each component of the mixture corresponding to the appropriate $\bar{\pi}_q$.

The preceding definitions are used by [6] to prove the consistency of an algorithm that partitions nodes based on their empirical degrees, in the global case where the whole network is studied. We need to redefine certain terms so they pertain to the local case. First, let $D_i^* := \sum_{j\in[n]\setminus\{e,i\}} A_{ie}A_{ij}A_{je}$, so that

$$T_i^* := \frac{D_i^*}{\sum_{i\in[n]\setminus\{e,i\}} A_{ie}}.$$

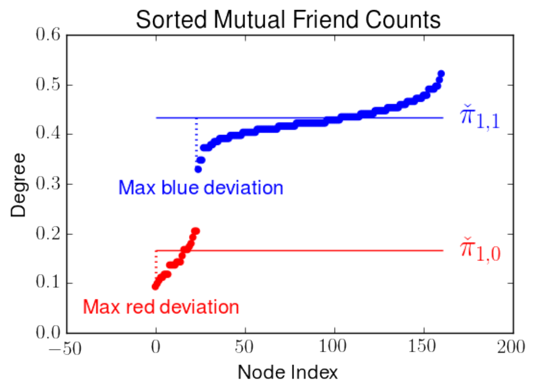

Figure 2: A sequence of (normalized) mutual friend counts. The red and blue horizontal lines mark the expected value for each community. The value of d_n^* in this case would be the max blue deviation (the blue deviation is greater than the red deviation).

Then,

$$d_n^* := \max_{s\in[Q]} \left[\sup_{i\in[n]\setminus\{e\}:Z_i=s,A_{ie}=1} |T_i^* - \breve{\pi}_{Z_e s}| \right],$$

where $\breve{\pi}_{rs} := \mathbb{P}(A_{ij} = 1 \mid A_{ie} = 1, A_{je} = 1, Z_e = r, Z_i = s)$. d_n^* is visualized in an example in Figure 2.

Remark 1. *$\breve{\pi}_{rs}$ can be expressed as*

$$\breve{\pi}_{rs} = \sum_{t\in[Q]} \pi_{st} \frac{\pi_{rt}\alpha_t}{\sum_{u\in[Q]} \pi_{ru}\alpha_u}.$$

3 Statistical Analysis of `Mutual-Friends`

The motivation for the main theorem comes from the following observation: clearly, if each mutual friend count is close enough to its corresponding $\breve{\pi}$, the gap between the ego's community and any other community should be easily found by using a one-dimensional clustering algorithm. [6] provides guarantees for this task when the $(Q-1)$-largest gaps are used to cluster the Q communities in the whole (not ego) network.

Theorem 1. *As long as $0 < t = O(\sqrt{\log n/n})$,*

$$\mathbb{P}(d_n^* > t) \overset{n\to\infty}{\longrightarrow} 0.$$

The theorem provides a guarantee for the concentration of the mutual friend counts. In order to use this theorem effectively, an additional assumption must be made on the values of $\breve{\pi}_{Z_e s}$. Of course, $\breve{\pi}_{Z_e Z_e}$ should not equal $\breve{\pi}_{Z_e s}$ for any community $s \neq Z_e$; otherwise, nodes in community s will be assumed to be in the same community as the ego. Furthermore, if $\breve{\pi}_{Z_e Z_e} < \breve{\pi}_{Z_e s}$ for any community $s \neq Z_e$, s will be taken to be the ego's community rather than Z_e.

3.1 Conditions on π and α

It turns out that the concerns in the preceding paragraph are easily avoided. To see this, assume for simplicity of notation that the block matrix π is of the form $\pi_{ss} = w$ and $\pi_{rs} = b$ for all $w \neq s$, i.e. all of the within-block probabilities are w, and all of the between-block probabilities are b (similar results can be found without this assumption, but to keep this document short we omit them). This means that

$$\breve{\pi}_{Z_e s} = \frac{1}{\bar{\pi}_{Z_e}} \left[\left(\sum_{t\in[Q]\setminus\{Z_e,s\}} \pi_{Z_e t}\pi_{st}\alpha_t \right) + \pi_{Z_e s}(\pi_{Z_e Z_e}\alpha_{Z_e} + \pi_{ss}\alpha_s) \right]$$

and

$$\breve{\pi}_{Z_eZ_e} = \frac{1}{\bar{\pi}_{Z_e}} \left[\left(\sum_{t \in [Q] \setminus \{Z_e\}} \pi^2_{Z_e t} \alpha_t \right) + \pi^2_{Z_e Z_e} \alpha_{Z_e} \right].$$

Then, the necessary and sufficient conditions for the ego's community to have the highest expected mutual friend count are

$$\begin{cases} \breve{\pi}_{Z_eZ_e} - \breve{\pi}_{Z_e1} > 0 \\ \vdots \\ \breve{\pi}_{Z_eZ_e} - \breve{\pi}_{Z_eQ} > 0 \end{cases} \iff \begin{cases} (w-b)(w\alpha_{Z_e} - b\alpha_1) > 0 \\ \vdots \\ (w-b)(w\alpha_{Z_e} - b\alpha_Q) > 0 \end{cases} \iff \begin{cases} \alpha_{Z_e} > \frac{b}{w}\alpha_1 \\ \vdots \\ \alpha_{Z_e} > \frac{b}{w}\alpha_Q. \end{cases}$$

Assuming an assortative SBM (i.e. $w > b$), the ego community can be quite small in the overall network. Often, w is taken such that $w \gg b$, so that the ego community will generally be identifiable from the mutual friend counts. Note that even if $\alpha'_{Z_e} = \alpha'_s$ for some $s \neq Z_e$, the mutual friend counts are still expected to be higher for the ego's community, since $\pi_{Z_eZ_e} > \pi_{Z_es}$. This property endows the `Mutual-Friends` algorithm with a kind of robustness, as long as the whole network can be adequately modeled with an SBM.

3.2 Clustering the Mutual Friend Counts

Once the mutual friend counts have been obtained, any reasonable one-dimensional clustering algorithm should identify the ego's community, up to the above conditions. As mentioned earlier, in [6], a $(Q-1)$-largest gaps algorithm is analyzed. In this local clustering case, finding the single-largest gap should, in many cases, identify the ego community. In practice, k-means with $k = 2$ is more robust to random deviations from the cluster means. The simulation results describe this in more depth.

3.3 Perfect Recovery of the Within-community Set

The necessary and sufficient conditions for $\breve{\pi}_{Z_eZ_e} > \breve{\pi}_{Z_es}$ to hold are described above, but these conditions do not guarantee that the `Mutual-Friends` algorithm will perform well if the clustering is done by taking the largest gap in the mutual friend counts. In fact, even if the mutual friend counts are clustered perfectly (with zero variance) around their cluster means, there is still a way of setting the parameters π and α such that the largest-gap algorithm will fail to return the correct ego community. The following results provide a value for t so that the bound in Theorem 1 is guaranteed to bound the probability of the largest-gap algorithm making a mistake.

Remark 2. *The value of t which defines the condition that the largest-gap algorithm makes zero errors is*

$$t < \frac{\frac{\breve{\pi}_{Z_eZ_e} - \max_{q \neq Z_e} \breve{\pi}_{Z_eq}}{2} - \frac{\max_{q \neq Z_e} \breve{\pi}_{Z_eq} - \min_{q \neq Z_e} \breve{\pi}_{Z_eq}}{2}}{2}.$$

Corollary 1. *If*

$$\breve{\pi}_{Z_eZ_e} - \max_{q \neq Z_e} \breve{\pi}_{Z_eq} > \max_{q \neq Z_e} \breve{\pi}_{Z_eq} - \min_{q \neq Z_e} \breve{\pi}_{Z_eq},$$

then as $n \to \infty$, the probability of the largest-gap algorithm making any mistake goes to zero.

3.4 Simulated Comparisons to Spectral Clustering

Although the theoretical results show that the `Mutual-Friends` algorithm works quite well even for smaller networks, competing algorithms also boast strong performance. To verify the practical utility of the algorithm, we compare it to a localized variant of spectral clustering on the graph Laplacian, which is the most common algorithm used for graph clustering. Spectral clustering takes the number of communities as a parameter, K; however, in this case, we cannot use $K = Q$, the number of communities in the SBM, since there may be fewer than Q communities in the ego network. Hence, we use $K = 2$ inthe simulations.

For these simulations, we sample graphs from different SBMs. In all simulations, $\pi_{qq} = 0.1$ for all q and $\pi_{qr} = 0.01$ for all $q \neq r$. This corresponds to the fully symmetric block matrix with

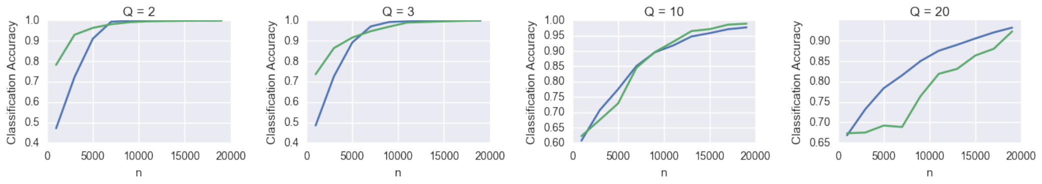

Figure 3: Blue lines depict the accuracy of `Mutual-Friends`. Green corresponds to spectral clustering on the graph Laplacian with $K = 2$. When the number of clusters is small (e.g. $Q = 2, 3$), `Mutual-Friends` performs worse than spectral clustering until the network is large enough, after which it slightly outperforms spectral clustering. Once $Q = 10$, `Mutual-Friends` and spectral clustering perform similarly. When the number of communities increases to $Q = 20$, `Mutual-Friends` consistently outperforms spectral clustering. Figure 4 contains the FPR and FNR plots.

$w = 0.1$ and $b = 0.01$. We vary Q, to depict the relationship between the two methods when there are few communities in the network vs. when there are many communities. As Q varies, we set $\alpha = (1/Q, \ldots, 1/Q)$; this corresponds to balanced community sizes. Figure 3 summarizes these results. The simulated whole-network sizes are $n = 500, 1000, 2000, \ldots, 20000$, with every 1000-node increment between 1000 and 20000. For each combination of network size n and number of communities Q, 50 samples are drawn from the associated SBM. The 50 accuracy values for each method are then averaged and connected to create the plots. The corresponding plots for the false positive rate (FPR) and false negative rate (FNR) are given in Figure 4.

The simulations show that `Mutual-Friends` performs similarly to spectral clustering for small numbers of communities. However, as the number of communities increases, `Mutual-Friends` begins to outperform spectral clustering, and in general, `Mutual-Friends` appears to outperform spectral clustering for large network sizes. These results do not come as a surprise, as spectral clustering is only guaranteed to perform well when the correct number of communities is given. In particular, we suspect that ego networks from SBMs with large values of Q have a “hairy” ego community node cluster, in the sense that stray nodes (having $Z_i \neq Z_e$) with extremely sparse connections to the ego's node cluster are increasingly prevalent when Q increases. These nodes may not be discernable from the ego's community to spectral clustering, but they have low mutual friend counts and thus are correctly classified by `Mutual-Friends`.

Accuracy is not the only metric for comparison: after all, we are after an algorithm which can reliably learn about the graph local to some node as quickly as possible. The runtime of `Mutual-Friends` is governed by two factors: first is how long the graph database takes to return the mutual friend counts of a given node; second is how long it takes to run one-dimensional clustering on the counts. The first task is assumed to be quite fast, especially in comparison to general graph queries, as this information is likely to be cached. One-dimensional clustering can be done in linear time by finding the largest gap in the unsorted vector of mutual friend counts.

This is in contrast to most implementations of spectral clustering which require constructing the adjacency matrix and graph Laplacian. In fact, our implementation of the `Mutual-Friends` algorithm in Python is typically at least an order of magnitude faster than the SciPy implementation of spectral clustering [13]. Furthermore, there are currently no established guarantees (statistical or not) for spectral clustering when it is used for community detection on ego networks.

Figure 4 demonstrates that `Mutual-Friends` matches the performance of spectral clustering on large networks. As for FPR, spectral clustering performs better when $Q = 2$, possibly because we are setting the number of communities to $K = 2$ for spectral clustering. For larger values of Q, `Mutual-Friends` is either comparable to or outperforms spectral clustering.

4 Recovering the Entire Ego Community

We introduced `Mutual-Friends` as a primitive from which to build more complex procedures. One such procedure is recovering the entire set of nodes belonging to the same community as the ego node. Several modified PageRank algorithms essentially perform this task: they take as input some starting node and use a biased random walk to cut the graph around this node [4, 16]. Each community of

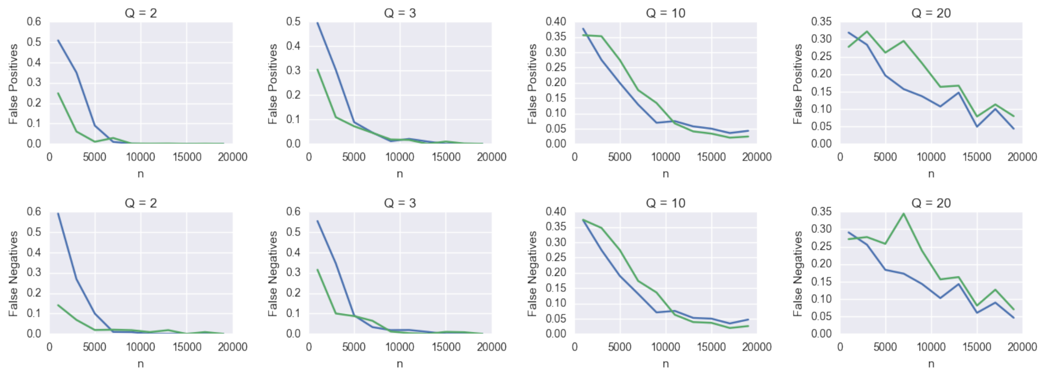

Figure 4: In the top four plots, blue lines depict the FNR (proportion of incorrect classifications out of the nodes that are truly outside the ego community) of `Mutual-Friends`. Green corresponds to spectral clustering on the graph Laplacian with $K = 2$. Lower is better. The bottom four plots are the same but they depict the FPR (proportion of incorrect classifications out of the nodes that are truly inside the ego community).

a stochastic block model essentially acts as an Erdös-Renyi graph. The diameter of such graphs is known to be quite small (this is true of real-world networks as well).

This fact suggests the naïve procedure outlined in Algorithm 2 for discovering the whole ego community using `Mutual-Friends`. Within 20 iterations of `Mutual-Friends` on a graph with 2000 nodes drawn from an SBM, we are able to recover approximately 98% of all of the nodes in the same community as the ego node in the graph. For reference, only about 22% of all of the nodes in the same community as the ego node are actually contained in the ego network, so with a small number of iterations of the `Mutual-Friends` algorithm we are able to recover several times more nodes in the same communtiy as the ego node. Figure 5 illustrates several runs of Algorithm 2.

Algorithm 2: Recoving the entire community that the ego belongs to

```
Data: Ego node $e$; number of steps to take $N$
Result: Ideally, the maximal set of nodes $S$ such that $Z_i = Z_e$ and for all $i \in S$
begin
    $\mathfrak{V} \longleftarrow \{e\}$;
    $(D_i^*)_i \longleftarrow$ neighbors$(e)$;
    $S \longleftarrow$ Mutual-Friends$(e, (D_i^*)_i)$;
    for $i \in [N]$ do
        $\mathcal{V} \sim$ Unif$(S \setminus \mathfrak{V})$;
        $\mathfrak{V} \longleftarrow \mathfrak{V} \cup \{\mathcal{V}\}$;
        $(D_i^*)_i \longleftarrow$ neighbors$(\mathcal{V})$;
        $S \longleftarrow S \cup$ Mutual-Friends$(\mathcal{V}, (D_i^*)_i)$;
    end
    return $S$;
end
```

5 Conclusions

`Mutual-Friends` is introduced as a competing method within the local graph clustering literature. It utilizes the sampling bias observed in edge-induced subgraphs to obtain a good clustering result for large networks with many communities that can be modeled as an SBM. Although it solves a sub-problem of the one solved by modified PageRank methods, Algorithm 2 is proposed as a simple extension which competes directly with such methods.

Although `Mutual-Friends` has good theoretical properties and compares favorably to standard spectral-based techniques, a challenge to applying `Mutual-Friends` to real social network data is

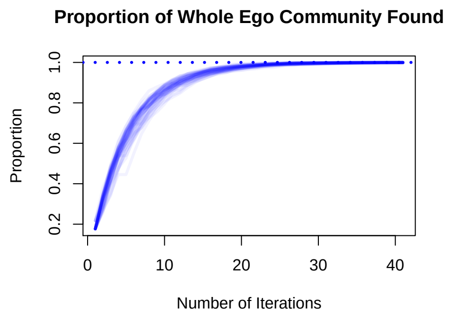

Figure 5: Each line represents a single run of Algorithm 2 for increasing iteration number N. All runs were performed on a single sample from an SBM with parameters $w = 0.2$ and $b = 0.01$ with $\alpha = (0.5, 0.5)$.

that social networks are not generally able to be modeled with a standard SBM since users do not generally belong to one community only. This means that the mutual friend counts will not be clearly separated and the algorithm will not detect an ego community.

Mixed membership SBMs (MMBs) [2] are a natural generalization of the standard SBM, wherein nodes, in some sense, belong to a mixture of communities. Having a local theory of such a model is therefore naturally of interest. The `Mutual-Friends` algorithm is essentially limited to using one-hop information about the graph, which is a severe reduction compared to utilizing the full graph topology. There may be a way of incorporating some limited knowledge of the local graph connectivity (such as some two-hop data) to improve the resolution of the method just enough to reliably learn the more complicated mixed-membership model.

Furthermore, we use standard spectral clustering with $K = 2$ as the comparison method. It especially does not perform as well as `Mutual-Friends` when the number of communities is large. To understand why should require a better understanding of the top two eigenvectors when Q is especially large. Furthermore, the choice of $K = 2$ is to make the comparison to `Mutual-Friends` fair as it is a fully nonparametric procedure. However, a better choice of K will likely lie between 2 and the number of unique communities observed in the ego network (and it is possibly not equal to this upper bound).

Acknowledgments

We thank Cosma Shalizi, Brian Junker, Jing Lei, Neil Spencer, Alden Green, Momin Malik, and the rest of Stat-Networks group at Carnegie Mellon for many valuable discussions throughout the development of this work. This work was partially supported by AFOSR grant #FA9550-14-1-0141.

References

[1] Emmanuel Abbe, Afonso S Bandeira, and Georgina Hall. Exact recovery in the stochastic block model. *IEEE Transactions on Information Theory*, 62(1):471–487, 2016.

[2] Edoardo M Airoldi, David M Blei, Stephen E Fienberg, and Eric P Xing. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, 9(Sep):1981–2014, 2008.

[3] Reid Andersen, Christian Borgs, Jennifer Chayes, John Hopcraft, Vahab S Mirrokni, and Shang-Hua Teng. Local computation of pagerank contributions. In *International Workshop on Algorithms and Models for the Web-Graph*, pages 150–165. Springer, 2007.

[4] Reid Andersen, Fan Chung, and Kevin Lang. Local graph partitioning using pagerank vectors. *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, 2006.

[5] Christian Borgs, Michael Brautbar, Jennifer Chayes, Sanjeev Khanna, and Brendan Lucier. The power of local information in social networks. In *International Workshop on Internet and Network Economics*, pages 406–419. Springer, 2012.

[6] Antoine Channarond, Jean-Jacques Daudin, and Stéphane Robin. Classification and estimation in the stochastic blockmodel based on the empirical degrees. *Electronic Journal of Statistics*, 6:2574–2601, 2012.

[7] Wen-Yen Chen, Yangqiu Song, Hongjie Bai, Chih-Jen Lin, and Edward Y Chang. Parallel spectral clustering in distributed systems. *IEEE transactions on pattern analysis and machine intelligence*, 33(3):568–586, 2011.

[8] Kimon Fountoulakis, Xiang Cheng, Julian Shun, Farbod Roosta-Khorasani, and Michael W Mahoney. Exploiting optimization for local graph clustering. *arXiv preprint arXiv:1602.01886*, 2016.

[9] Jing Lei and Alessandro Rinaldo. Consistency of spectral clustering in stochastic block models. *The Annals of Statistics*, 43(1):215–237, Feb 2015.

[10] Yixuan Li, Kun He, David Bindel, and John Hopcroft. Overlapping community detection via local spectral clustering. *arXiv preprint arXiv:1509.07996*, 2015.

[11] Julian Mcauley and Jure Leskovec. Discovering social circles in ego networks. *ACM Transactions on Knowledge Discovery from Data*, 8(1):1–28, Feb 2014.

[12] Lorenzo Orecchia and Zeyuan Allen Zhu. Flow-based algorithms for local graph clustering. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1267–1286. Society for Industrial and Applied Mathematics, 2014.

[13] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[14] Karl Rohe, Sourav Chatterjee, and Bin Yu. Spectral clustering and the high-dimensional stochastic blockmodel. *Ann. Statist.*, 39(4):1878–1915, 08 2011.

[15] Julian Shun, Farbod Roosta-Khorasani, Kimon Fountoulakis, and Michael W Mahoney. Parallel local graph clustering. *arXiv preprint arXiv:1604.07515*, 2016.

[16] Daniel A. Spielman and Shang-Hua Teng. A local clustering algorithm for massive graphs and its application to nearly-linear time graph partitioning. *CoRR*, abs/0809.3232, 2008.

[17] Lu Wang, Yanghua Xiao, Bin Shao, and Haixun Wang. How to partition a billion-node graph. *2014 IEEE 30th International Conference on Data Engineering*, Mar 2014.

[18] Donghui Yan, Ling Huang, and Michael I Jordan. Fast approximate spectral clustering. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 907–916. ACM, 2009.

[19] Yunpeng Zhao, Elizaveta Levina, and Ji Zhu. Community extraction for social networks. *Proceedings of the National Academy of Sciences*, 108(18):7321–7326, 2011.

Proof of Theorem 1

The proof of Theorem 1 requires a concentration-type bound for T_i^*, the normalized mutual friend count. This is possible by the following lemma, regarding the unnormalized mutual friend count, D_i^*.

Lemma 1.

$$D_i^* := \sum_{j\in[n]\setminus\{e,i\}} A_{ie}A_{ij}A_{je} \Bigg| Z_e, Z_i = s, \{A_{\ell e}\}_{\ell \neq i}, A_{ie} = 1 \sim \text{Bin}\left(\sum_{j\in[n]\setminus\{e,i\}} a_{je}, \breve{\pi}_{z_e s}\right).$$

This result depends on a rather intricate set of conditional random variables and events. Several conditions must be met in order for mathematically-useful statements about ego neighbors to be made: the block membership of each neighbor should be known (as usual), but the block membership of the ego node itself should also be conditioned on (much of the behavior of the ego network is dictated by the ego node's own block membership); in addition, the event that the neighbors are connected to the ego node has to be conditioned on.

Proof of Lemma 1. For each term in the sum, observe that

$$\begin{aligned}\mathbb{P}\left(A_{ij}A_{je} = 1 | Z_e, Z_i = s, \{A_{\ell e}\}_{\ell\neq i}, A_{ie} = 1\right) &= \begin{cases} 0, & A_{je} = 0 \\ \mathbb{P}\left(A_{ij} = 1 | Z_e, Z_i = s, \{A_{\ell e}\}_{\ell\neq i}, A_{ie} = 1\right), & A_{je} = 1\end{cases} \\ &= \begin{cases} 0, & A_{je} = 0 \\ \mathbb{P}\left(A_{ij} = 1 | Z_e, Z_i = s, A_{je}, A_{ie} = 1\right), & A_{je} = 1\end{cases} \\ &= \begin{cases} 0, & A_{je} = 0 \\ \breve{\pi}_{Z_e s}, & A_{je} = 1.\end{cases}\end{aligned}$$

In particular, this implies that $\{A_{ij} | Z_e, Z_i, \{A_{\ell e}\}_{\ell\neq i}\}_j$ for all $j \in [n] \setminus \{e, i\} : A_{je} = 1$ are identically and independently distributed (since they are d-separated) as Bernoulli random variables with probability $\breve{\pi}_{Z_e s}$.

The distribution of D_i^* conditional on the above conditioning set is therefore

$$\begin{aligned}\mathbb{P}\left(D_i^* = k | Z_e, Z_i = s, \{A_{\ell e}\}_{\ell\neq i}, A_{ie} = 1\right) &= \mathbb{P}\left(\sum_{j\in[n]\setminus\{e,i\}} A_{ij}A_{je} = k \Bigg| Z_e, Z_i = s, \{A_{\ell e}\}_{\ell\neq i}, A_{ie} = 1\right) \\ &= \mathbb{P}\left(\sum_{j\in[n]\setminus\{e,i\}:A_{je}=1} A_{ij} = k \Bigg| Z_e, Z_i = s, \{A_{\ell e}\}_{\ell\neq i}, A_{ie} = 1\right),\end{aligned}$$

which, by the preceding paragraph, is binomial with the desired parameters. □

We also use the fact that as the network grows, the number of nodes in each community will concentrate. We denote by B_q the number of nodes in the ego network which are in community q:

$$B_q | \{Z_i\}_i \overset{iid}{\sim} \text{Bin}\left(n_q := \sum_{i\in[n]\setminus\{e\}} 1_{z_i = q}, \pi_{z_e q}\right).$$

Let N_q be the (random) number of nodes in community q, and n_q be realization of this random variable. Then, the simultaneous concentration of all community sizes within the ego network is

$$\mathcal{B} := \bigcap_{q\in[Q]} \left\{|B_q - N_q\pi_{Z_e q}| \leq \sqrt{n \log n}\right\}.$$

Lemma 2. *The probability that at least one block fails to concentrate, conditioned on the block assignments, goes to zero as $1/n^2$. In other words,*

$$\mathbb{P}\left(\mathcal{B}^C | \{Z_i\}_i\right) \sim \frac{1}{n^2}.$$

Proof of Lemma 2. The probability that at least one block fails to concentrate according to $\mathcal{B}$ is:

$$\begin{aligned}
\mathbb{P}\left(\mathcal{B}^C|\{Z_i\}_i\right) &= \mathbb{P}\left(\bigcup_{q\in[Q]}\left\{|B_q - N_q\pi_{Z_eq}| > \sqrt{n\log n}\right\}\middle| Z\right)\\
&\leq \sum_{q\in[Q]}\mathbb{P}\left(|B_q - N_q\pi_{Z_eq}| > \sqrt{n\log n}\middle| Z\right)\\
&\leq \sum_{q\in[Q]} 2\exp\left\{-2\frac{n\log n}{n}\right\}\\
&= 2\frac{Q}{n^2}.
\end{aligned}$$

□

Proof of Theorem 1.

$$\mathbb{P}\left(d_n^* > t\right) \leq \mathbb{E}\left[\mathbb{P}\left(d_n^* > t|\mathcal{B}, z\right)\mathbb{P}\left(\mathcal{B}|Z\right) + \mathbb{P}\left(\mathcal{B}^C|Z\right)\right].$$

By Lemma 2, the second term in the expectation goes to zero independently of t, since it is not a function of t. Hence,

$$\begin{aligned}
\mathbb{P}\left(d_n^* > t\right) &\leq \mathbb{E}\left[\mathbb{P}\left(d_n^* > t|\mathcal{B}, Z\right)\right] + o(1)\\
&= \mathbb{E}\left[\mathbb{P}\left(d_n^* > t|\mathcal{B}, Z, \{A_{\ell e}\}_\ell\right)\right] + o(1)\\
&= \mathbb{E}\left[\mathbb{P}\left(\bigcup_{s\in[Q]}\ \bigcup_{i\in[n]\setminus\{e\}:Z_i=s,A_{ie}=1}\{|T_i^* - \check{\pi}_{Z_es}| > t\}|\mathcal{B}, Z, \{A_{\ell e}\}_\ell\right)\right] + o(1)\\
&\leq \mathbb{E}\left[\sum_{s\in[Q]}\ \sum_{i\in[n]\setminus\{e\}:Z_i=s,A_{ie}=1}\mathbb{P}\left(|T_i^* - \check{\pi}_{Z_es}| > t|\mathcal{B}, Z, \{A_{\ell e}\}_\ell\right)\right] + o(1)\\
&= \mathbb{E}\left[\sum_{s\in[Q]}\sum_{i\in[n]\setminus\{e\}} 1_{Z_i=s}A_{ie}\mathbb{P}\left(|T_i^* - \check{\pi}_{Z_es}| > t|\mathcal{B}, Z, \{A_{\ell e}\}_\ell\right)\right] + o(1)\\
&= \sum_{\{a_{\ell e}\}_\ell\in\{0,1\}^{n-1}}\ \sum_{z\in[Q]^n}\sum_{s\in[Q]}\sum_{i\in[n]\setminus\{e\}} 1_{z_i=s}a_{ie}\mathbb{P}\left(|T_i^* - \check{\pi}_{Z_es}| > t|\mathcal{B}, Z, \{A_{\ell e}\}_\ell\right)\mathbb{P}\left(\{A_{\ell e}\}_\ell, Z\right) + o(1)\\
&= \sum_{\{a_{\ell e}\}_\ell}\sum_{s\in[Q]}\sum_{i\in[n]\setminus\{e\}}\sum_{z_e\in[Q]} a_{ie}\mathbb{P}\left(|T_i^* - \check{\pi}_{Z_es}| > t|\mathcal{B}, Z_e, Z_i = s, \{A_{\ell e}\}_\ell\right)\mathbb{P}\left(\{A_{\ell e}\}_\ell, Z_e, Z_i = s\right) + o(1)\\
&= \sum_{s\in[Q]}\sum_{z_e\in[Q]}\sum_{i\in[n]\setminus\{e\}}\sum_{\{a_{\ell e}\}_\ell} a_{ie}\mathbb{P}\left(|T_i^* - \check{\pi}_{z_es}| > t|\mathcal{B}, Z_e, Z_i = s, \{A_{\ell e}\}_{\ell\neq i}, A_{ie} = 1\right)\mathbb{P}\left(\{A_{\ell e}\}_{\ell\neq i}, A_{ie} = 1, Z_e, Z_i = s\right) + o(1)\\
&\leq \sum_{\{a_{\ell e}\}_\ell}\sum_{s\in[Q]}\sum_{z_e\in[Q]}\sum_{i\in[n]\setminus\{e\}} a_{ie}2\exp\left\{-2\sum_{q\in[Q]}\left(n\pi_{z_eq} - \sqrt{n\log n} - 1\right)t^2\right\}\mathbb{P}\left(\{A_{\ell e}\}_{\ell\neq i}, A_{ie} = 1, Z_e, Z_i = s\right) + o(1).
\end{aligned}$$

We can simplify $\mathbb{P}\left(\{A_{\ell e}\}_{\ell\neq i}, A_{ie} = 1, Z_e, Z_i = s\right)$ as:

$$\begin{aligned}
\mathbb{P}\left(\{A_{\ell e}\}_{\ell\neq i}, A_{ie} = 1, Z_e, Z_i = s\right) &= \mathbb{P}\left(A_{ie} = 1|Z_e, Z_i = s\right)\mathbb{P}\left(\{A_{\ell e}\}_{\ell\neq i}|Z_e, Z_i = s\right)\mathbb{P}\left(Z_e, Z_i = s\right)\\
&= \pi_{z_es}\left(\prod_{j\in[n]\setminus\{e,i\}}\bar{\pi}_{z_e}^{a_{je}}\left(1 - \bar{\pi}_{z_e}\right)^{1-a_{je}}\right)\mathbb{P}\left(Z_e\right)\mathbb{P}\left(Z_i = s\right)\\
&= \pi_{z_es}\left(\prod_{j\in[n]\setminus\{e,i\}}\bar{\pi}_{z_e}^{a_{je}}\left(1 - \bar{\pi}_{z_e}\right)^{1-a_{je}}\right)\alpha_{z_e}\alpha_s\\
&= \pi_{z_es}\left(\bar{\pi}_{z_e}^{\sum_{j\in[n]\setminus\{e,i\}}a_{je}}\left(1 - \bar{\pi}_{z_e}\right)^{n-2-\sum_{j\in[n]\setminus\{e,i\}}a_{je}}\right)\alpha_{z_e}\alpha_s.
\end{aligned}$$

Going back to the bound of $\mathbb{P}\left(d_n^* > t\right)$, we make the substitution $k = \sum_{j\in[n]\setminus\{e,i\}} a_{je}$:

$$\begin{aligned}
\mathbb{P}\left(d_n^* > t\right) &\le \sum_{\{a_{\ell e}\}} \sum_{s\in[Q]} \sum_{z_e\in[Q]} \sum_{i\in[n]\setminus\{e\}} a_{ie} 2\exp\left\{-2\left(\sum_{q\in[Q]} n\pi_{z_e q} - \sqrt{n\log n} - 1\right)t^2\right\} \pi_{z_e s}\left(\bar{\pi}_{z_e}^k (1-\bar{\pi}_{z_e})^{n-2-k}\right)\alpha_{z_e}\alpha_s + o(1) \\
&= \sum_{k=0}^{n-1} \binom{n-1}{k} \sum_{s\in[Q]} \sum_{z_e\in[Q]} (k+1)2\exp\left\{-2\left(\sum_{q\in[Q]} n\pi_{z_e q} - \sqrt{n\log n} - 1\right)t^2\right\} \pi_{z_e s}\bar{\pi}_{z_e}^k (1-\bar{\pi}_{z_e})^{n-2-k}\alpha_{z_e}\alpha_s + o(1) \\
&= 2\sum_{s\in[Q]} \sum_{z_e\in[Q]} \frac{\pi_{z_e s}\alpha_{z_e}\alpha_s}{1-\bar{\pi}_{z_e}} \exp\left\{-2\left(\sum_{q\in[Q]} n\pi_{z_e q} - \sqrt{n\log n} - 1\right)t^2\right\}(1+(n-1)\bar{\pi}_{z_e}) + o(1),
\end{aligned}$$

which converges to 0 as $n \to \infty$ for all fixed $t > 0$ as well as $t = O(\sqrt{\log(n)/n})$. Notably, all of the constants are known in this bound; the additional asymptotic term introduced by Lemma 2 is used for convenience, but its constants are known as well. □